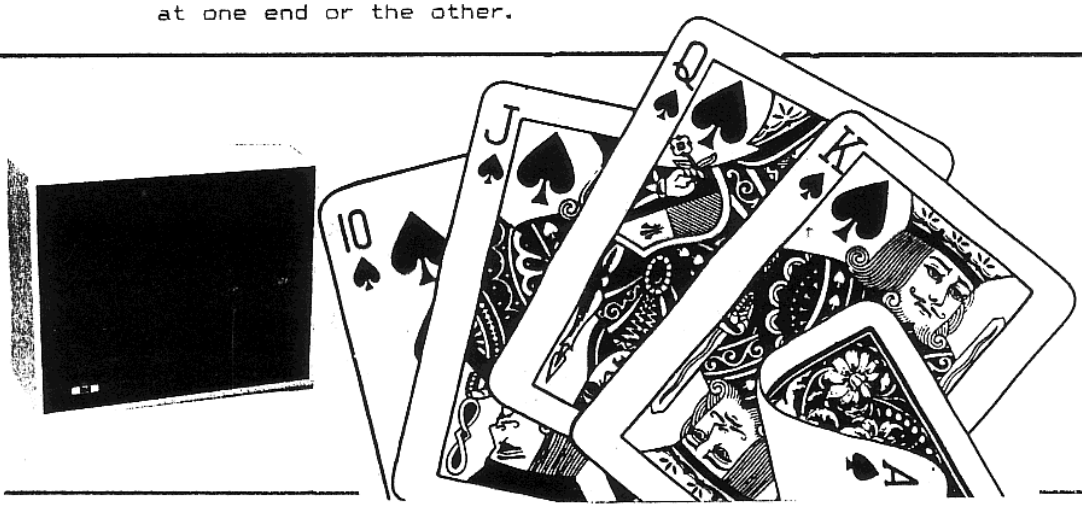
SORTING

When you have grown tired of playing games on your computer and have no particular use for your word-processor, data base, spread sheet and other utilities, there is an exercise you may like to try. If you can invent a quicker method of sorting lists of either numbers into numerical order, or words into alphabetical order, you could become famous. Some such inventors have even gone into tinkering with the binary coding in the Inner Sanctum of computers. It is very useful to know how the most popular sorts of sorts actually work. Here we will attempt to explain the various popular methods of sorting and to assist my explanations you will need the complete suit from a pack of playing cards, 13 cards from Ace to King. So while you are looking for a pack of cards and taking out one complete suit, I will be figuring out the best way of keeping the explanations simple. What is daunting is that I have read explanations in books and magazines and have never found one that is truly comprehensible. Anyhow, here is my effort...

PUSH-DOWN SORT

This Sort is extravagant in both time and memory space. It should be used for short lists only. Generally, but not always it is used to sort entries as they are INPUT. Each entry is compared with the preceding entries and put into order between RETURN and INPUT. Often it is necessary to enter the number of items to be entered at the beginning. For short lists it is a good system because the sorting time is hidden between the INPUTs. However, as you will see, the sorting time gets progressively longer and longer and longer.

Make sure your 13 cards are in random order then place one face up, on the table. Being only one card it is already sorted. Place the second card on the table to the right of the first card. If card no.2 is a higher card than no.1 then swap them over, otherwise leave them as they are. Place card no.3 to the right of card no.2. Now compare card no.3 with card no.1 and swap if necessary and go back to card no.3 and compare it with no.2 and swap if necessary. Now go to card no.2 and compare it with card no.1 and, again, swap if necessary. The three cards should now be in numerical order. You've guessed it. Put card no.4 to the right of no.3. Compare and swap (if necessary) card no.4 with 1, then 2, then 3. Go back to card no.3 and do a comparison check with 1, then 2. Back again to no.2 amd make a comparison check with no.1. Phew! Do you get the feeling that you are repeating yourself. Told you it was extravagant. However, continue, one new card at a time. Eventually you will arrive at a fully sorted suit of cards. You should have made the Ace a low card; a ONE, but it doesn't matter. It'll end up at one end or the other.



PUSH-DOWN SORT

```
                                          PUSH-DOWN SORT  example.
A MICRO MAGIC program. August 1986.
100 CLEAR2000:DIMZ(100)                 Dimensioned for 100 items.
110 INPUT"HOW MANY ITEMS";L              Number of items to be sorted.
120 FORN=1TOL-1                            -1 as LOOP is FROM 0 (NOT 1).
130 PRINT"ITEM"N+1;:INPUT"ENTER NUMBER";Z(N)
140 IFN=0THEN160                           We cannot sort just one item.
150 GOSUB200                               To the SORTING sub-routine.
160 NEXT                                   Collect the next entry.
170 FORP=0TON-1:PRINTZ(P):NEXT          Print the sorted list.
180 GOTO100                                'ave another go.

200 FORI=JTO0STEP-1:FORJ=0TON-1
210 IFZ(J)>Z(J+1)THENTEMP=Z(J):Z(J)=Z(J+1):Z(J+1)=TEMP
220 NEXT:NEXT:RETURN
```

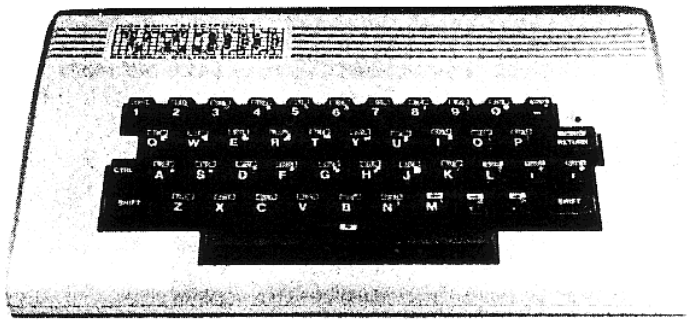
The above sort can be used after all the entries have been entered. The sorting routine would simply draw upon the stored items one by one. The reason why it is called a PUSH-DOWN SORT is because information is said to be in a STACK and the information is drawn from the stack on the principle of *last in first out*. As an analogy, using your 13 cards, stack them with only the top card showing. Now try a PUSH-DOWN SORT. Place the top card by the side of the stack. You can now compare it with the top card of the stack. Swap them over if the top card of the stack is a higher card. The only way you can continue is by beginning a third stack and then comparing the top card of the 1st stack with those on the top of the other two stacks. The final stack of sorted cards is seen to be, being pushed down. That's all, but you should now see why this sort of sort is extravagant in memory space as well as in sorting time.

WARNING My VZ-200s contain bugs. Yours too, possibly. Not yet having a VZ-300 I do not know if one particular bug is in those VeeZeds also. If a loop such as FORI=1TO13:PRINTI:NEXT is run you would expect that **I** would end up as **13**. But on my VZ-200s **I** ends up, for some unaccountable reason, as **14**. To check your computer try this test loop:

```
FORI=1TO13:PRINTI;:NEXT:PRINT"I="I
```

If you too get **I= 14** then your computer has caught the bug. In a Sorting program the Variable used to count the number of entries may need to be reduced by 1, otherwise you may be trying to make a swap with a entry that doesn't exist and an ERROR MESSAGE will appear in a line where no error can be found.

Two more points before going onto another SORT system. If the Sorting program is dealing with whole numbers, then all the variables should have a PERCENT sign after them. Like **Z%(N)**, but you knew that didn't you. The second point is that often you will see in a Sorting progtam a DUMMY item which may look like this: Z$(0)="ZZZ". This would be sorted down to the end of the list. This is very useful as it can be used to indicate when a list is ended. The ZZZ can produce STOP TAPE, both when Tapesaving and Retrieving. Then there would be no need to keep a record of how many items are on the tape.



BUBBLE SORT

BUBBLE SORT is also called RIPPLE SORT or a title with the word WAVE in it. This is the most popular system. You most likely already have programs with this sort of sort in it. MICRO DATA is one such program. This time, in addition to the suit of playing cards you will also need a coin. Lay the 13 cards in random order in a row on the table. Place your coin below the leftmost card, card no.1. Compare the COINCARD with its neighbor on its right. If the COINCARD is a higher card than its neighbor then swap them over. Otherwise leave them as they are. Now move the coin up to the 2nd card (which may be the same card as before if you made a swap). Then compare this COINCARD with its righthand neighbor and again swap if necessary. Continue moving and comparing until the coin has reached the 12th position. By this time the KING will have reached 13th place. The ACE is a 1. Take the coin back to the beginning again and begin another PASS. This time the QUEEN will arrive at position 12 when the coin reaches the 11th position. Keep going, reducing the PASS by one each time, until all the cards are sorted. See, it was quite easy, after all.

In computer language a PASS would look like this...

```
FORI=1TO12:IFZ$(I)>Z$(I+1)THENSWAP:NEXT
```

I know there is no command in a VeeZed called SWAP, but there is a short routine which does the same thing. The main thing, at this point, is that we will need an OUTER loop to control this one and to reduce the PASS by one each time. How about this...

```
FORJ=12TO0STEP-1
FORI=0TOJ-1
IFZ$(I)>Z$(I+1)THENT$=Z$(I):Z$(I)=Z$(I+1):Z$(I+1)=T$
NEXT:NEXT:RETURN
```

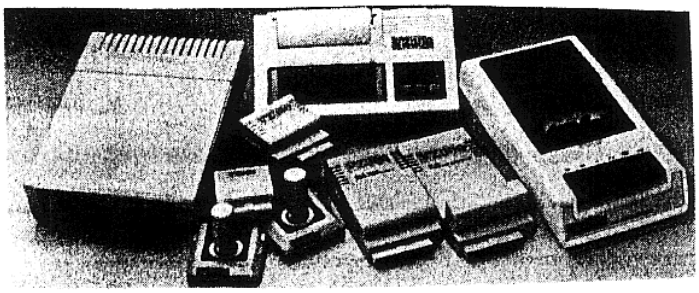
You will see that the OUTER loop, of these Nested Loops, runs backwards from 12 to 0. The INNER loop, therefore, being J—1 must run for 12 the first time, then for 11, then for 10 and so on. Just what we wanted, eh?

Also notice that we have used a TEMPorary String Variable in which to park the data so that it won't get lost while we do a swap. Being a sub-routine RETURN goes back to PRINT the sorted list. Here then is an EXAMPLE PROGRAM...

```
                                   BUBBLE SORT
A MICRO MAGIC program. August 1986.
100 CLEAR2000:DIMA$(100):N=1
110 INPUT"NAME";A$(N):IFA$(N)=""THEN120ELSEN=N+1:GOTO110
120 N=N-1
130 FORP=1TON-1:FORI=1TON-P
140 IFA$(I)>A$(I+1)NEXTT$=A$(I):A$(I)=A$(I+1):A$(I+1)=T$
150 NEXT:NEXT

200 FORI=1TON:PRINTA$(I):NEXT
210 PRINT"END OF LIST":PRINT
220 GOTO100
```

When finished entering items to be listed just press RETURN without making an entry. The blank String Variable will be automatically cancelled.



SHELL SORT

Nothing to do with Mary-Anne sorting shells by the sea shore. A Mr. D. Shell devised this one. And he must have been either a gambler or a student of the Laws of Chance. He considered that if he tinkered with the random list before Bubble Sorting them, the quicker it would run. By doing a sort of organised random sort he would increase the chances of the final sort zipping through the list like streaked lightning. He was right. Then bigger the list the better the chances. So here we habe a sort that is suited for very long lists.

You will need your 13 cards and TWO coins. Any coins will do. IBM owners use $1 coins while VeeZedders make do with one cent pieces. Lay the cards in a row in random order. It will help if you put the KING somewhere about the middle of the row. Place a coin below the leftmost card, card no.1. This will be called the COINCARD. Now for the rules according to Mr Shell....

Count the cards - 13. Divide by 2 (ignore any remainder) - 6
The 6th card from the left is called the LINK.
The number 6 is called the LINK NUMBER.
Use your other coin to mark the space between the 6th and 7th card.

ADD LINKNUMBER to COINCARD. 6 + 1 = 7.
Now ADD the 7 to the LINK NUMBER. 6 + 7 = 13
PUSH UP cards 1, 7 and 13. We would also push up 14 but there isn't a card 14. We only have 13.
The cards pushed up form the first CHAIN.
BUBBLE SORT the three cards.
In case you've forgotten how, here's what to do...

Compare and sort cards 1 and 7.
Compare and sort cards 1 and 13.
Compare and sort cards 7 and 13.
Here ends the 1st CHAIN.
Push the cards back in line.

Move the coin from card no.1 to card no.2. This is the new COINCARD.
ADD LINKNUMBER to COINCARD 2 + 6 = 8.
PUSH UP cards 2 and 8. As before we cannot push up card 14 as there isn't one.
Compare and sort 2 and 8.
That ends the 2nd Link.
Push the cards back in line.

Don't fret. It will get more exciting in a moment.

Move the coin to card no.3. This is the new COINCARD.
ADD LINKNUMBER to the COINCARD. 6 + 3 = 9.
PUSH UP 3 and 9.
Compare and sort 3 and 9.
End of 3rd LINK.
Push the cards back in line.

Continue moving the coin and comparing and sorting as above until you have completed the 6th CHAIN. That is when the coin reaches card 6.

Take the coin back to 1.
Divide the LINKNUMBER by 2 (ignoring any remainder) 6 ÷ 2 = 3.
LINKNUMBER is now 3.
You will be adding by 3s this time.
Place 2nd coin between cards 3 and 4.
PUSH UP cards 1, 4, 7, 10 and 13.
Compare and sort 1 & 4, 1 & 7, 1 & 10, 1 & 13.
Compare and sort 4 & 7, 4 & 10, 4 & 13.
Compare and sort 7 & 10, 7 & 13.
Compare and sort 10 & 13.
PUSH DOWN the cards. Move the coin to no.2.
PUSH UP cards 2, 5, 8 and 11.
Compare and sort 2 & 5, 2 & 8, 2 & 11.
Compare and sort 5 & 8, 5 & 11.
Compare and sort 8 & 11.

PUSH DOWN the cards. Move the coin to 3 (the LINKCARD).
PUSH UP 3, 6, 9 and 12.
Compare and sort 3 & 6, 3 & 9, 3 & 12.
Compare and sort 6 & 9, 6 & 12.
Compare and sort 9 & 12.
PUSH DOWN the cards.
BUBBLE SORT

The coin has reached the LINKCARD so it goes back to no.1. Dividing the LINKNUMBER by 2 (ignoring any remainder) = 1, so with the coin back at no.1, compare no.1 with no.2, swap if necessary. Then coin on no.2, compare and sort 2 & 3, move coin, 3 & 4, move coin, 4 & 5 and all the way up until the coin reaches card 12. Then from 1 to 11, 1 to 10 etc. until the last PASS is done, which will be 1 & 2.

All we need now is a program that will do all that. In fact I got so carried away with the success of the first example, I prepared 2 examples. The first READs Playing Cards from DATA. PRINTs the list so that you can see the cards changing places. A SOUND indicates the beginning of the final BUBBLE SORT. And, believe me, it's fast.

NOTE: The words in DATA are preceded by single letters to allow them to be sorted in alphabetical order, and each item is 8 characters long. The words used for variables relate to the explanation above.

```
0 '******EXAMPLE OF SHELL SORT***********MICRO MAGIC*************
100 CLEAR3000:LIMIT=100:DIMZ$(LIMIT):CLS
110 DATAE    FIVE,B     TWO,J     TEN,A     ACE,H   EIGHT
114 DATAM    KING,C   THREE,D   FOUR,I   NINE,F     SIX
118 DATAK    JACK,G   SEVEN,L   QUEEN
130 FORI=1TO13:READZ$(I):NEXT
140 LIMIT=I-1:LCARD=INT(LIMIT/2):LNUMB=LCARD+1
155 'SHELL SORT
160 GOSUB300
170 A=1
180 FORCOIN=ATOLCARD
190 FORCHAIN=LCARDTOLIMITSTEPLNUMB
200 IFZ$(COIN)>Z$(CHAIN)THENGOSUB400
210 IFCHAIN>LIMITTHENLCARD=INT(LNUMB/2):GOTO220
220 NEXT
230 A=A+1
240 NEXT
255 'BUBBLE SORT
260 FORP=1TOLIMIT
262 FORI=1TOLIMIT-P
270 IFZ$(I)>Z$(I+1)THENT$=Z$(I):Z$(I)=Z$(I+1):Z$(I+1)=T$
280 NEXT
290 NEXT:SOUND10,1:GOSUB300
299 GOTO299
300 PRINT@0,"LIST":FORI=1TO13:PRINTZ$(I):NEXT:SOUND22,2:RETURN
400 TEMP$=Z$(COIN)
410 Z$(COIN)=Z$(CHAIN)
420 Z$(CHAIN)=TEMP$
430 GOSUB300
440 RETURN
```

The second example is along the same lines as the first, only this time the items to be listed are INPUT by you. If, instead of typing in an item, you just hit RETURN, the program will go into the sorting routines automatically. The unsorted list is printed and then replaced by the sorted list.

```
0 '******EXAMPLE OF SHELL SORT***********MICRO MAGIC*************
100 CLEAR3000:LIMIT=1000:DIMZ$(LIMIT):I=1:CLS
110 PRINT"ENTER WORD"I;:INPUTZ$(I):IFZ$(I)=""THEN130
120 I=I+1:GOTO110
130 A=1:LIMIT=I-1:LCARD=INT(LIMIT/2):LNUMB=LCARD+1:CLS:GOSUB300
140 FORPASS=1TOLIMIT:FORCOIN=1TOLCARD
150 FORCHAIN=LCARDTOLIMITSTEPLNUMB
160 IFZ$(COIN)>Z$(CHAIN)THENGOSUB400
170 IFCHAIN>LIMITTHENLCARD=INT(LNUMB/2):GOTO180
180 NEXT:A=A+1
190 NEXT
200 LCARD=INT(LIMIT/2):LNUMB=LCARD-1:IFLCARD=2THEN220
210 NEXT
220 FORP=1TOLIMIT:FORI=1TOLIMIT-P
230 IFZ$(I)>Z$(I+1)THENT$=Z$(I):Z$(I)=Z$(I+1):Z$(I+1)=T$
240 NEXT
250 NEXT:SOUND10,1:CLS:GOSUB300
260 GOTO260
300 FORI=1TO13:PRINTZ$(I):NEXT:SOUND22,2:PRINT"LIST ENDS":RETURN
400 TEMP$=Z$(COIN):Z$(COIN)=Z$(CHAIN):Z$(CHAIN)=TEMP$:RETURN
```

INSERTION SORT

Here is a program where the sorting is actually done by the operator. As Z$(I) from 1 to 12 are the months of the year, then the I used to store other information is made the same. This means that if Z$(I)="JULY" the Z(I) will be a 7. Extra sub-routines can be added to allow for changing the information set against each month, or perhaps, link two Strings together. This way a Bubble Sort could be used to sort either the TITLE, in this case the MONTHS, or sort the associated information so that you can either list by TITLE or list by the other information. If the TITLE was the name of a piece of music and the other information was the ARTIST, the list could be printed in alphabetical order of TITLE or ARTIST.

Programming with kind of facility in mind you should become familiar with 2-dimensional arrays. Those that look like Z$(A,B) where the A holds the position in a list and the B holds the information to be listed. The variations are limited only by your imagination. And as we said at the beginning, the VeeZed computer is ideal for those who take a delight in programming in BASIC. The editing of lines is easy, trial and error of ideas are supported by error codes that aid in getting it right. All you really need is the time.

```
0 '******INSERTION SORT*********MICRO MAGIC PROGRAM**AUGUST 1986
100 CLEAR2000:I=12:DIMZ$(I),Z(I)
110 CLS:GOSUB300:PRINT"ENTER SALES FIGURE FOR "
120 FORI=1TO12
130 PRINTZ$(I);:PRINTTAB(12)"$";:INPUTZ(I)
140 NEXT:CLS
150 PRINT"SELECT MONTH":INPUTM$:I=VAL(M$):IFM$=""THEN150
160 S$=MID$(M$,1,3)
162 IFS$="OCT",I=10ELSEIFS$="NOV",I=11ELSEIFS$="DEC",I=12
164 IFS$="JUL",I=7ELSEIFS$="AUG",I=8ELSEIFS$="SEP",I=9
166 IFS$="APR",I=4ELSEIFS$="MAY",I=5ELSEIFS$="JUN",I=6
168 IFS$="JAN",I=1ELSEIFS$="FEB",I=2ELSEIFS$="MAR",I=3
170 IFI<0ORI>12THEN:GOTO150
180 PRINTZ$(I),:PRINT"$"Z(I):I=0
190 PRINT:GOTO150
300 FORI=1TO12:READN$:Z$(I)=N$:NEXT:RETURN
302 DATAJANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE,JULY,AUGUST
304 DATASEPTEMBER,OCTOBER,NOVEMBER,DECEMBER
400 'IF A NUMBER IS INPUT THEN I=VAL(M$) MAKES I THAT NUMBER
401 'OTHERWISE MID$(M$,1,3) TAKES THE 1ST THREE LETTERS OF
402 'WHATEVER MONTH HAS BEEN TYPED IN AND THEN MAKES I THE
403 'MONTH NUMBER. LESS THAN 0 OR MORE THAN 12 IS REJECTED.
404 '*****REMOVE ALL REMS*****************************************
```